- What does OSS need to learn from the same?

Position Papers

A total of 17 position papers are included in the workshop proceedings, which are available online and discussed in the full workshop report (see **Further Information** below). The position paper titles and authors are as follows:

- *Software Architectures and Open Source Software – Where can Research Leverage the Most?* by Budi Arief, Cristina Gacek and Tony Lawrie
- *Configuration Management for Open Source Software* by Ulf Asklund and Lars Bendix
- *Software Engineering Lessons from Open Source Projects* by Jai Asundi
- *Creating a Free, Dependable Software Engineering Environment for Building Java Applications* by Marcus Bittman, Robert Roos and Gregory M. Kapfhammer
- *Open Source Development: An Arthurian Legend* by Jonathon E. Cook
- *The ramp-up challenge in open-source software projects* by Davor Cubranic
- *Corporate Source: Applying Open Source Concepts to a Corporate Environment* by Jamie Dinkelacker and Pankaj K. Garg
- *Software Engineering Research in the Bazaar* by Ahmed E. Hassan, Michael W. Godfrey and Richard C. Holt
- *Open Source Software: The Other Commercial Software* by Scott A. Hissam and Charles B. Weinstock
- *Conceptual Sociological Model for Open Source Software* by Kouichi Kishida
- *Reputation Layers for Open-Source Development* by Hassan Masum
- *Taxonomy of Open Source Software Development* by Kumiyo Nakakoji and Yasuhiro Yamamoto
- *Open Source Software Developments in XP Style* by Yoshiyuki Nishinaka
- *Introducing a "Street Fair" Open Source Practice Within Project Based Software Engineering Courses* by Dan Port and Gail Kaiser
- *Software Development Practices in Open Software Development Communities: A Comparative Case Study* by Walt Scacchi
- *Leveraging Open-Source Communities To Improve the Quality and Performance of Open-Source Software* by Douglas C. Schmidt and Adam Porter
- *Open Source Development: A suitable method to introduce a standardized communication protocol?* by Achim Spangler

Presentations

The workshop consisted of four 90-minute discussion sessions. Each session was focused on the issues raised by a mini-presentation based on one of the position papers. The four mini-presentations were:

- *Leveraging Open-Source Communities To Improve the Quality and Performance of Open-Source Software* (Schmidt and Porter)
- *Open Source Development: An Arthurian Legend* (Cook)
- *Corporate Source: Applying Open Source Concepts to a Corporate Environment* (Dinkelacker and Garg)
- *Taxonomy of Open Source Software Development* (Nakakoji and Yamamoto).

Further Information

The complete proceedings, agenda, and presentation slides from *Making Sense of the Bazaar: 1st Workshop on Open Source Software Engineering* are permanently housed at http://opensource.ucc.ie/icse2001/. The full workshop report will be published in early 2002 as part of a special issue of *IEE Proceedings – Software* (v.149:1) on Open Source Software Engineering. The full report will contain summary descriptions of the 17 submitted position papers as well as a discussion of the issues raised in the course of the workshop.

Meeting Challenges and Surviving Success: The 2nd Workshop on Open Source Software Engineering is currently being planned for 2002.

Report on Metrics 2001: The Science & Practice of Software Metrics Conference

Annabella Loconsole, Daniel Rodriguez,
Jürgen Börstler, Rachel Harrison
Umeå University, Sweden
Reading University, UK
{bella, jubo}@cs.umu.se
{d.rodriguez-garcia,r.harrison}@reading.ac.uk

Abstract

This paper reports on the IEEE 7th International Software Metrics Symposium (METRICS 2001), held in London, England, from the 4th to 6th April 2001 and co-hosted with the 12th European Software Control and Metrics conference (ESCOM 2001). Metrics Symposia are the premier event in the area of software metrics and attract most of the leading researchers and industrial practitioners.

Keywords: Conference Report, Metrics 2001

Introduction

METRICS 2001 attracted people from 22 different countries including most of the leading figures in this area. The program general chair was Dieter Rombach (Fraunhofer Institute for Experimental Software Engineering –IESE-, Germany), the program chair was Claes Wohlin (Blekinge Institute of Technology, Sweden) and Adrian Cowderoy (Professional Spirit Ltd, UK) was the coordinator for the ESCOM-Metrics conferences.

The program presented covered theory and practice of software development. The authors presented papers on measurement, empirical studies (evaluating new technologies, product based meas-

urement related to software architectures, COTS and the web, evolving systems, formal experiments using engineers and students) quantitative and qualitative approaches applied to software development, management and quality assurance. There were 2 parallel sessions, a panel, and 3 keynote speakers.

Keynotes of METRICS 2001 were given by Prof. Anneliese Amschler Andrews (Colorado State University) for the opening of the symposium. Tom DeMarco (The Atlantic Systems Guild, Inc.) speech closed the ESCOM 2001 conference and was open to METRICS 2001 participants. Finally, Dave G. Griffiths (BT Labs) closed the symposium.

The Invited Talks

Learning to Measure or Measure to Learn?

Anneliese Amschler Andrews, Colorado State University, USA

Anneliese emphasised the use of empirical studies and she analysed how the symposiums have changed over the time (for example, in the first Software Metrics Symposium in 93, there was only one paper related to empirical studies). While in the past the focus was on defining new metrics, the focus now is on learning from what we have defined.

In software engineering research area, it is necessary to have more empirical work, more validation, and techniques from multiple angles. Software engineers need to be flexible, and borrow from other disciplines. Some lessons to learn are:

4) Adapt the tools to the situation.
5) Look at the problem from multiple angles.
6) Determine which data is relevant.
7) It is necessary to use multiple methods.

The Agile Organization (Paul Rook memorial lecture)

Tom DeMarco, The Atlantic Systems Guild, Inc

This speech, based on the speaker's book [2], was the ESCOM closing session and METRICS 2001 participants were invited.

According to Tom DeMarco, nowadays change is difficult for organizations because their people are too busy *-over-optimised-*; activities must be done in a hurry and with fewer people (6 persons now do the work that once was done by 13). This means that organizations are probably more efficient but less flexible. Therefore, they cannot change and in turn perform faster in the future. He suggested that organizations need *slack* i.e., degree of freedom in time, budget manpower space etc necessary to make change possible. His *agility* principle stated that *business is not the same as busy ness* and he suggest for the future:

1) "Less" efficiency. More slack for the future, more reflection and slow development in order to allow change.
2) Light processes. The processes have become bigger but DeMarco is in favour of thin processes such as XP, Crystal methods, SCRUM, and Dynamic System Developing Methodology.
3) Management and prioritisation. The decision to add an extra project makes the duration of the existing projects in the organization even longer.
4) Invest in human capital.

Software Engineering - What Has It Done for You?

David G. Griffiths, BT Labs, UK.

David's talk was based on the article "*The Future of Software*" [1]. David defended that long-term research is an investment and some characteristics and criteria for long- and short-term research were explained. Although his talk was quite focused on technology, he discussed what will influence the future based on the concept that the use of software changes over time. This implies that software will need to be adaptive and some of the discussed technologies include:

- Agents, mobile agents, agent communication, etc. Among these technologies, BT Zetus (an open source project) was briefly explained.
- Software will be more intelligent, auto configurable and adaptive, better interfaces, etc.
- Better networks and intelligent networked houses.

During the final questions, there was a bit of controversy defining long and short-term research. David defined long-term research around 3 to 5 years. Other attendees such as Pfleeger objected that 3 years might not be a long-term research and it looks like this period is influenced by the fact that companies need a quick return on investments.

The Sessions

Session 1: Software Effort and Cost

Building a Software Cost Estimation Model Based on Categorical Data. L. Angelis, I. Stamelos, and M. Morisio.

The authors analysed the possibilities to generate multi-organisational software cost estimation model through the analysis of data collected by the international standard benchmarking group (ISBSG). Start-up companies can use the model without much experience in cost estimation. It can also be used to produce estimates for new projects or for benchmarking purpose. The generated model behaved better than conventional least squares regression models using the same database. The results cannot be compared with the results obtained in other studies where categorical data is used because they were conducted on different cost data set.

Using Public Domain Metrics to Estimate Software Development Effort. R. Jeffery, M. Ruhe, and I. Wieczorek.

This presentation investigated the accuracy of cost estimation when modelling techniques are applied to a large scale industrial data set. The first issue addressed in this study was the selection of a technique in a given context. The second issue discussed was the assessment of the feasibility of using multi-organisational data compared to the benefits from company specific data collection. As a result of this study, some modelling techniques performed better with ISBSG data while some other performed better when using the company's own data.

Predicting with Sparse Data. M. Shepperd and M. Cartwright.

In this presentation the speaker described the Sparse Data Method

to predict project cost related factors. The method was based on a pair wise comparison technique and the "Saaty's Analytic Hierarchy Process". A tool called "Data Salvage" supported the sparse data method. This approach added value to expert judgement producing more precise results in effective prediction of project costs related factors.

Session 2: Software Inspections

A Controlled Experiment to Assess the Effectiveness of Inspection Meetings. A. Bianchi, F. Lanubile, and G. Visaggio.

The authors carried out an experiment with more than 100 students to compare the effectiveness of inspections meetings. They compared real teams against nominal teams, which are the ones that do not interact face to face. They discovered that real teams do not surpass the nominal teams. Team meetings do not create a substantial amount of synergisms and the performance is not related to the duration of the meetings.

Investigating the Impact of Reading Techniques on the Accuracy of Different Defect Content Estimation Techniques. B. Freimut, O. Laitenberger, and S. Biffl.

The authors performed an experiment with 169 students to analyse the impact of reading techniques on the accuracy of defect content estimation techniques (DCET). In particular, they concluded that scenario-based reading (algorithmic guidelines) and checklist-based reading have no significant effect on the estimation of defects. Therefore, they concluded that (i) defect estimation technique can be used independently of the reading technique; (ii) subjective estimation is quite accurate and can be used when no other DCET is applied.

Analysing the Influence of Team Size and Defect Detection Technique On the Inspection Effectiveness of a Nominal Team. S. Biffl and W. Gutjahr.

Based on the data of the experiment described above, the authors proposed a statistical model to compute inspection benefits, net gain and return of investment. The authors concluded that: (i) a mix of reading techniques is more effective than any team that uses only the most suitable reading technique for a single inspector; (ii) larger teams find more defects but they are also less effective and finally (iii) the cost-benefit model limit the optimal team size.

Session 3A: Software Projects

Assessing the Benefits of Imputing ERP Projects with Missing Data. I. Myrtveit, U. Olsson, and E. Stensrud.

The speaker discussed ways to handle missing and incomplete data, which is a well-known problem in empirical software engineering. Some methods are discussed and then applied to estimate effort in enterprise resource planning (ERP) systems. Among the imputation methods presented (which replace missing values by suitable estimates), some sampling methods waste less information than the listwise deletion method. However, these sampling methods cannot correct affected data.

A Fuzzy logic Based Set of Measures for Software Project Similarity: Validation and Possible Improvements. A. Idri and A. Abran.

The speaker proposed a set of new measures based on fuzzy logic to measure the software project similarity attribute. These new measures are especially useful when the software project attributes are described by an ordinal or nominal scale (through the use of linguistic variables such as "low" and "high"). These similarity measures were validated using a set of 4 axioms. Then they were validated empirically using the COCOMO'81 database.

An Experiment for Evaluating the Effectiveness of Using a System Dynamics Simulation Model in Software Project Management Education. D. Pfahl, N. Koval, and G. Ruhe.

Pfahl described an experiment to investigate the effectiveness of using a process simulation model for computer-based training in Software Management. The experiment is based on System Dynamics (SD) simulation model where the students learned software management principles through scenarios. The experiment used a pre-test-post-test to demonstrate how the treatment involving SD had a positive impact on the tests results.

Session 3B: Software Modules

A Consideration of the Impact of Interactions with Module Effects on the Direct Measurement of Subjective Software Attributes. J. Moses.

Based on two datasets, Moses used a Bayesian inference procedure to assess whether the consistency of measurement of a modular attribute may be influenced by module effects. For example, module length interacts with the chance of correctly classifying maintainability and cohesion.

Measuring Coupling and Cohesion of Software Modules: An Information Theory Approach. E. Allen, T. Khoshgoftaar, and Y. chen.

The authors proposed coupling and cohesion measures based on Information Theory. The information theory approach has the benefit of being sensitive to patterns. The authors also presented a case study comparing the information-theory approach to software metrics with the count-based approach.

Cohesion is Structural, Coherence is Functional: Different Views, Different Measures. V. B. Mišić.

Mišić analysed the cohesion of a module looking into its functionality instead of its internal structure. The author observed that cohesion is an external property rather than an internal one when analysing some definitions of cohesion (e.g. Fenton and Pfleeger, define cohesion as the extend to which the individual components of a module are needed to perform the same task). Mišić gives the definitions and properties of this new approach.

Session 4A: Panel Session

What Good Are Metrics? The Views of Industry and Academia. Panel chair: S. L. Pfleeger, Systems/Software, Inc.

The panellist discussed some of the following questions: Why should we use metrics? Are they good for business? Do they really tell us something about quality and productivity? Are the predictions right? How can we match the reality and the vision? Among

the questions discussed we can highlight the following ones:

What do the practitioners think about the metrics created by academy? Too much focus on complex metrics and their theoretical demonstrations are academic weaknesses. On the contrary, companies are more focused on operational, practical metrics. According to attendees from companies, successful metrics should be very simple, easy to understand, low cost, and its implementation should not be very sophisticated.

What academies do right and wrong? The education in software engineering should be improved, but while it is easy to find Ph.D. students in for example, robotics, the students interested in software metrics are few. One possibility is to change the student's curriculum and teach them software modelling and design. People from industry have tried to organise courses on quality management but the classes are usually empty while other technical courses e.g. Java would be more popular. According to an attendee a person with a Java course on his/her curriculum vitae would obtain higher salary than another person having a quality management course. Companies' responsibility should be to employ people with quality management skills in addition to technical skills.

Are there metrics that businesses need the ones that researchers are producing? It is difficult to say what are good metrics because they are usually context dependent. Metrics are sometimes misleading because data is usually collected from projects where information is available. People are more focused on process measurement rather than on project measurement while there is a need of project measurements.

Session 4B: Testing

Usage Measurement for Statistical Web Testing and Reliability Analysis. C. Kallepally, and J. Tian.

Based on log files generated by Web servers, authors applied their Unified Markov Models (UMM) for statistical high-level testing. In these UMM usage information is represented as transition probabilities associated with different state transitions. In addition, they show how the log information can feed reliability models. This is explained through a case study using their university Web server.

An experiment on Lead-Time Impact in Testing of Distributed Real-Time Systems. T. Olsson, N. Bauer, P. Runeson, and L. Bratthal.

The authors conducted an experiment to evaluate how system monitoring improves the testing process in a Distributed Real Time System. The experiment showed the efficiency and effectiveness of using system monitoring traces for identifying sources of failures compared to node-monitoring traces.

Understanding and Measuring the Sources of Variation in the Prioritisation of Regression Test Suites. S. Elbaum, D. Gable, and G. Rothermel.

The authors presented an empirical study to understand and quantify the sources of variation influencing different testing prioritisation techniques. The experiment was conducted on a large set of metrics. They defined the average percentage of faults metrics. They defined the average percentage of faults detected (APFD) as an independent variable, and a large number of dependent variables related to program metrics and change metrics. As a result, they discovered that some of the metrics were redundant in their information content while others were best predictors of APDF.

Session 5A: State-of-the-Art Seminars

Challenges of Global Software Development A. Mockus, and J. D. Herbsleb.

The goal of the speech was to present advantages, disadvantages and open questions in global software development (GSD) in which the activities are performed in a global setting. There are several problems in distributed software development; one of the biggest is the communication across sites. Project participants have usually different backgrounds, training, and languages. There are few opportunities to interact with other colleagues in remote sites and working hours overlap is very little. One way to diminish the problems encountered in distributed project setting is de-coupling of work, which involves adapting the development process, choosing suitable models and criteria, dividing the work in a modular structure. Another way to reduce the GSD problems is the creation of virtual sites that involves tools, practices and processes. When creating virtual sites it is mainly important to set up a common development environment, and to provide the tools for joint work sessions.

A Primer on Object-Oriented Measurement. K. El Emam.

Kaled presented an overview of object-oriented metrics (OOM) proposed in the last decade. They can be classified from a number of perspectives: static and dynamic, which measure cognitive complexity (test coverage is an example of a dynamic metric). OOM can also be defined at different levels, i.e., system level, file level, class level, and method level. They can be characterised according to the development phase and can be useful for both quality estimation and risk management. Finally, Kaled presented heuristics for empirical validation studies of metrics.

Session 5B: Maintenance and Evolution

Controlling Overfitting in Software Quality Models: Experiment with Regression Trees and Classification. T. M. Khoshgoftaar, E. B. Allen, and J. Deng.

The speaker showed an application of regression tree algorithm in order to classify modules and detect fault prone modules. The authors studied a large telecom system and investigated the minimum deviance and minimum node size parameters that affect the growth of the tree. The results was a strong correlation between minimum deviance and overfitting, i.e., a model is accurate when using the training dataset but it is less accurate when applied to other datasets. However, the effect of the minimum node size is not clear.

Defining and Applying Metrics in the Context of Continuing Software Evolution. J.F. Ramil, and M.M. Lehman.

Ramil presented a framework for measurement of software evolution covering the top-down, bottom-up, and mixed approaches. The authors applied the CUSUM (Cumulative Sum) test on a

suite of 8 activity metrics (such as *ModulesCreated*, *ModulesChanged*, etc). This was done with the goal of examine the changes in productivity during a period of 11 years. Ramil concluded that 6 out of the 8 indicators support the hypothesis of declining productivity after a period of approximately 10 months.

Evaluating Software Degradation through Entropy. A. Bianchi, D. Caivano, F. Lanubile, and G. Visaggio.

The authors presented the concept of *software systems entropy*, which is defined as a class of metrics to assess the degree of disorder in a software system. They also defined Model Dependency Descriptor (MDD). MDD models software artefacts as a graph of software lifecycle objects connected by internal and external traceability relationships. The authors validated their approach against 3 direct metrics (no. of discovered defects, no. of slipped defects and maintenance effort) using scheduling systems developed by students as a case study. They found that the measure of entropy is consistent with the direct metrics but it was affected by the level of granularity defined by MDD.

Session 6A: Software Measurement Infrastructure

A Targeted Assessment of the Software Measurement Process. M. Berry, and M. F. Vandenbroek.

In this speech it was described the meta-Measurement framework used to make assessments. The framework was applied to obtain a qualitative evaluation of the relationship of measurements with the key process area Project Tracking and Oversight (PTO). The speaker underlined the importance of qualitative studies in software measurement. The assessment was conducted with a web-based survey, which exploited the experience and attitude of employers to assess the relationship between measurement and a specific software process. The experiment identified opportunities to improve measurement support for PTO and practices of PTO that can be improved.

V-GQM: A Feed-Back Approach to Validation of a GQM Study. T. Olsson, and P. Runeson.

In his talk Thomas presented weaknesses of the Goal/Question /Metrics (GQM). Then he introduced the V-GQM (Validating GQM) as a method for validating the study after data collection. Questions and metrics are analysed with the purpose of adapting them to the current environment and to gain knowledge for later studies. Finally, the speaker presented a case study of the V-GQM performed in an industrial context. One of the attendee (R. van Solingen) objected that there are no weaknesses in the GQM method.

Integrating Goal-Oriented Measurement in Industrial Software Engineering: Industrial Experience with and Additions to the Goal/Question/Metric Method (GQM). R. van Solingen, and E. Berghout.

Rini presented an extension of the initial idea of GQM method. One addition concerns the establishment of frequent feedback sessions; through these sessions the team can learn, i.e., gaining knowledge and changing behaviour. The second addition regards a cost benefit analysis of the measurement program. A GQM project performed without these extensions would probably lead to failure.

Session 6B: Object-Oriented Systems and COTS

The Impact of Design Properties on Development Cost in Object Oriented Systems. L. C. Briand, and J. Wüst.

In this talk it was investigated the relationships between class size and its development effort taking into account complexity metrics such as coupling and cohesion. To do so, authors analysed an interactive musical editor using Poisson regression and regression trees to build cost prediction models. They concluded that accurate predictions can be obtained from the classes' interface. The addition of more complicated complexity measures do not improve cost predictions significantly.

Robustness and Diagnosability of OO Systems Designed by Contracts. B. Baudry, Y. Le Traon, and J. Jezequel.

The speaker defined measures for robustness and diagnosability of object oriented systems in a mathematical way when the *design-by-contract* technique is used. They presented the results of a case study to measure the contribution of designing by contract to the quality of the system. The results showed an improved diagnosability of the system

A Method for Efficient Measurement-based COTS Assessment and Selection – Method Description and evaluation Results. M. Ochs, D. Pfahl, G. Chrobok-Diening, and B. Nothhelfer-Kolb.

The authors presented CAP (COTS Acquisition Process) method to assess and evaluate COTS. The authors verified their approach performing two case studies during a period of 7 and 8 months respectively. Applying the GQM, authors proved its validity in several ways: (i) it was more efficient than brute force approaches, (ii) it represented a low additional cost and finally, (iii) it was economically beneficial.

Session 7A: Industrial Application and Experiences

Better Validation in a World-Wide Development Environment. C. Ebert, C. Hernandez Parro, R. Suttels, and H. Kolarczyk.

In this presentation, it was shown a case study to estimate 3 hypotheses in validation activities in global software development (GSD): 1) peer reviews should be located in one place 2). Coaching should be provided by peer engineers, 3) teams should be responsible of features independently from other teams and the various parts developed should be integrated continuously. These hypotheses would reduce the cost of non-quality. The 3 changes were implemented successfully in a global setting obtaining a cost reduction due to earlier defect detection and less defects introduced.

Measurement Automation: Methodological Background and Practical Solutions - A Multiple Case Study. S. Komi-Sirviö. P. Parviainen, and J. Ronkainen.

The speaker underlined the importance of doing goal oriented measurement with the lowest amount of effort. A software process improvement framework applied to support a measurement automation project was presented. The speaker showed two industrial examples where there was a need for measurement automation. The MetriFlame measurement management tool was used to sup-

port measurement automation in these industrial examples.

Making the Software Factory Work: Lessons from a Decade of Experience. H. P. Siy, J. D. Herbsleb, A. Mockus, M. Krishnan, and G. T. Tucker.

In this talk it was presented experiences on how to make a process centred software company work well. These experiences are based on interviews, questionnaires, assessments, and software metrics for medium-size companies. Among several lessons learned presented, the importance of separating roles in a software team was emphasised. However, everyone in the team should be responsible of process improvement activities but only a small group should lead and plan process efforts.

Session 7B: Prediction

Investigation of Logistic Regression as a Discriminant of Software Quality. N. F. Schneidewind.

Schneidewind investigated Logistic Regression Functions (LRF) when used in combination with Boolean Discriminant Functions (BDF), which are model for classifying quality. LRF are the probability of the occurrence of discrepancy reports on modules as a linear function of metrics, while BDF are Boolean functions to classify software modules into accepted or rejected. His conclusion were that (i) BDF are superior for quality discrimination; (ii) although LRFs are of limited value when used in isolation, the inspection cost is reduced compared when BDF are used alone; (iii) LRF can provide a ranking for identifying the quality of the modules.

Measurement, Prediction and Risk analysis for Web Applications. R. Fewster, and E. Mendes.

Firstly, the authors described a case study evaluation of student's implementation of Web sites, where metrics related to size, reusability, complexity, effort and confounding factors were collected. Second, the use of the Generalised Linear Model for effort prediction and risk management was discussed.

Using Simulation to Evaluate Prediction Techniques. M. Shepperd, and G. Kadoda.

The authors looked into the problem of choosing a prediction technique depending on the characteristics of the available dataset. Their approach was to generate simulated data as a way of controlling and generating a large number of validation cases. Then 3 prediction techniques were compared, namely stepwise regression, rule induction and case-based reasoning. Their conclusions suggested that there is a relationship between the technique used and the characteristics of the data. They provided some information about what circumstances favour a prediction technique.

Conclusions

In this paper, we have presented a summary of the IEEE 7th International Software Metrics Symposium including its keynotes and tutorial. Among recurrent issues raised on the symposium, we can highlight the following:

- A general tendency to be observed is the promotion of empirical research in software engineering. This is mainly due to the still apparent gap between industry and academic research. While the industry asks for metrics that are understandable, easy to manipulate and collect, academic researchers are more concentrated on metrics that can be proved formally. Therefore a closer collaboration between academic researchers and the industry is required.
- Some companies prefer goal-oriented process improvements to standardised ones, e.g. choosing a concrete area to improve in their process without the constraint of certifications.
- Long-term research is becoming a trend in industry, but the definition of "long-term research" in industry (3–5 years) differs from the academic definition where this period of time is considered a short-term.
- Thin processes are also gaining more and more popularity in industry.

References

[1] P. Brereton, D. Budgen, K. Bennett, M. Munro, P. Layzell, L. Macalulay, D. G. Griffiths, and C. Stannett. The Future of Software, *Communication of ACM*, Vol. 42, No. 12, Dec, 1999.

[2] T. DeMarco, *Slack: Getting Past Burnout, Busywork, and the Myth of Total Efficiency*, Broadway Books, 2001.

[3] *Proceedings of the 7th International Software Metrics Symposium*, IEEE Computer Society, 2001.

International Workshop on Software Configuration Management (SCM-10) New Practices, New Challenges, and New Boundaries

André van der Hoek
Department of Information and Computer Science
Institute for Software Research,
University of California Irvine
Irvine, CA 92697-3425 USA
andre@ics.uci.edu

Abstract

This report provides a brief summary of SCM-10, the ICSE 2001 10th International Workshop on Software Configuration Management. The primary goal of this workshop was to broaden the scope of SCM and establish ties with other disciplines that are strongly related to SCM—whether requiring some form of novel, advanced SCM functionality or influencing the field of SCM with newly available technology. As demonstrated by the accepted set of position papers and the lively discussion in the workshop, SCM-10 succeeded in achieving this goal and raised many new and important questions to be addressed in the years to come.

Keywords: Configuration Management